# Adaptive EDF: Using Predictive Execution Time

Kiyofumi Tanaka
School of Information Science
Japan Advanced Institute of Science and Technology (JAIST)
Email: kiyofumi@jaist.ac.jp

*Abstract*— **Along with the growing diversity and complexity of real-time embedded systems, significance of real-time scheduling is increasing. Rate monotonic (RM) and earliest deadline first (EDF) are representative scheduling algorithms for periodic tasks. RM has a merit that tasks with high-priority (short-period) have small jitters and short response times. However, it is impossible that processor utilization reaches 100% while maintaining schedulability or that tasks are given importance independent of their periods. On the other hand, EDF can utilize processors by 100% with schedulability, while it cannot give tasks fixed priorities or importance, therefore, it is difficult to keep jitters or response times of particular tasks short. This paper, in EDF algorithm, proposes a method of shortening response times of a task that is important for the system by introducing predictive execution times and dividing execution of the important task into two instances based on the predictive execution times. The use of predictive execution times has an effect of making deadline of the first instance earlier and therefore the first instance would be executed earlier by EDF. In the evaluation by simulation, the proposed method reduced average response times of the most important task by up to 14.2%.**

## I. Introduction

Along with the growing diversity and complexity of real-time embedded systems, real-time scheduling is becoming increasingly important. Control tasks that are required to completely meet real-time requirements (hard tasks) should be periodically invoked and be considered to spend their worst-case execution time (WCET). This is because the schedulability, that they satisfy their deadline requirements, must be confirmed in advance of system operation.

Rate monotonic (RM) and earliest deadline first (EDF) are representative scheduling algorithms for periodic tasks [1]. RM is one of fixed-priority algorithms and executes tasks with shorter periods preferentially. Although RM has a merit that tasks with high-priority (short-period) have small jitters and short response times, processor utilization cannot reaches 100% while maintaining schedulability. EDF is one of dynamic-priority algorithms and executes tasks with earlier absolute deadlines preferentially. Under EDF, processor utilization of 100% with schedulability can be achieved, while it is impossible to give tasks fixed priorities, which means it is difficult to keep jitters or response times of particular tasks short. This study explores algorithms based on EDF and aims to reduce response times of a particular task that is important for the system, while not affecting schedulability.

Complexity of current processors and programs makes the estimation of WCET difficult. For example, deep pipelining execution of machine instructions makes estimation of their execution times hard, or in a system with many tasks, whether each memory reference would hit in the cache memory or not is difficult to decide or predict [2]. In addition, the worst-case execution path in a program is almost impossible to trace since it includes many branches and loop structures, and all possible input patterns provide a vast search range [3]. Consequently, WCET is obliged to be pessimistically estimated and leads to having a large gap with actual execution times. Due to this gap, it is often difficult to obtain the best schedules in terms of response times or turnaround times by scheduling algorithms that take tasks' execution times into account.

In this research, a method of shortening response times of a particular task is proposed and evaluated. The method introduces predictive execution times in addition to WCET, divides execution of a particular task (most important for the system) into two instances, and gives the first instance an earlier deadline, which has a possibility of shortening the response time when the actual execution time is shorter than the planned WCET.

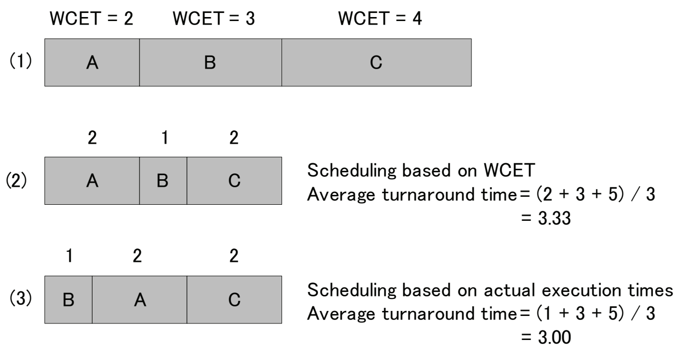
This paper consists of four sections. Section II proposes the adaptive EDF as an extended EDF algorithm. The proposed algorithm is evaluated by simulation in Section III. Section IV conclude this paper.

## II. The Adaptive Earliest Deadline First Algorithm

### A. Prediction of Execution Time

Generally, in real-time scheduling and its schedulability analysis, task execution is considered to spend its worst-case execution time (WCET). In practice, task execution time is unknown beforehand and therefore WCET must be supposed to spend, especially in hard real-time systems. However, in most cases, actual execution time is shorter than WCET. Since WCET is pessimistically estimated, the difference between the actual execution time and WCET tends to be large.

As the impact of the difference, for example, when SJF (Shortest Job First) algorithm is applied based on WCETs, the average turnaround time would be worse than that of the same algorithm based on actual execution times, although this is an oracle with prior information. In Fig. 1, (1) shows that WCETs of task A, task B, and task C are 2, 3, and 4, respectively, and the execution order is A, B, and then C based on SJF. When this order is applied to actual executions where actual execution times are 2, 1, and 2 for task A, task B, and task C, respectively, the average turnaround time becomes 3.33 (Fig. 1 (2)). On the other hand, if the actual execution times are known in advance and used in the algorithm, the execution order will be B, A, and then C, and the average turnaround time will be 3 under SJF as shown in Fig. 1 (3). Like this, decision based on WCETs is not necessarily optimal.



WCET = 2
WCET = 3
WCET = 4
(1)
A
B
C
2
1
2
(2)
A
B
C
Scheduling based on WCET
Average turnaround time = (2 + 3 + 5) / 3 = 3.33
1
2
2
(3)
B
A
C
Scheduling based on actual execution times
Average turnaround time = (1 + 3 + 5) / 3 = 3.00

Fig. 1. Scheduling based on Shortest Job First.

In EDF, schedulability analysis is based on tasks' execution times. If the sum of the processor utilization by all tasks does not exceed 100%, the task set is schedulable. In this analysis, it is assumed that tasks spend their WCETs. However, in actual system operation, tasks rarely spend their worst-case execution times. Rather, the actual execution times would be much shorter than the corresponding WCETs. Therefore, the processor utilization tends to be lower than that based on WCETs. In this research, a method of reducing response times by assuming predictive execution times, not WCETs, and setting tentative deadlines based on the predictive execution times is proposed.

In this strategy, execution times should be predicted. There are various possible ways to obtain the predictive execution times.

1) Random choice of execution times
2) Measurement in advance
3) Prediction using a history of execution

The above 1), random choice, has high possibility of choosing shorter execution times than actual execution times, and therefore would cause many deadline misses if a scheduling algorithm in which deadlines are decided depending on execution times is used. The second method, measurement in advance, seems effective but has a defect of not following the change of execution times when a task is executed multiple times in the system operation. The last one predicts execution times by an execution history of the same task and therefore can follow the fluctuation of execution times. For the present, this paper uses the following prediction method which corresponds to the above 3.

$$\begin{aligned} C_{i_{PET_k}} &= \alpha \times C_{i_{PET_{k-1}}} + (1-\alpha) \times C_{i_{ET_{k-1}}} \quad (1) \\ C_{i_{PET_0}} &= C_{i_{WCET}} \end{aligned}$$

Here, $C_{i_{PET_k}}$ means the predictive execution time for $k$th ($k = 0, 1, \ldots$) instance of a periodic task $\tau_i$. $C_{i_{ET_{k-1}}}$ is the execution time actually spent for the previous execution of the same task $\tau_i$. The initial value $C_{i_{PET_0}}$ is equal to WCET of the task ($C_{i_{WCET}}$). This formula calculates as the predictive execution time an weighted average of the previous predictive execution time and the previous actual execution time with the weighting coefficient $\alpha$ $(0 \leq \alpha \leq 1)$.

## B. Definition of the Adaptive EDF

In RM environments, it is possible to keep response times of important tasks short by giving short periods to the important tasks. However, it is impossible to set up the importance of tasks independent of the periods. On the other hand, EDF uses dynamic priorities and therefore cannot reflect static importance in the schedules. The adaptive EDF proposed in this paper introduces fixed importance of tasks independent of the tasks' periods.

The adaptive EDF targets a set of periodic tasks. Each task has a relative deadline that is equal to the period. In the adaptive EDF, an execution of an important periodic task is divided into two sub instances. That is, if $\tau_i$ is regarded as the most important task, the $k$th execution instance, $J_{i_k}$, of $\tau_i$ is divided into two parts, $J_{i_{PET_k}}$ and $J_{i_{REST_k}}$. $J_{i_{PET_k}}$ corresponds to the execution from the beginning of $J_{i_k}$ to the predicted finishing time. $J_{i_{REST_k}}$ corresponds to the execution from the predicted finishing time. Let the worst-case execution time of $\tau_i$ be $C_{i_{WCET}}$, the predictive execution time of $J_{i_k}$ be $C_{i_{PET_k}}$, and the execution time of $J_{i_{REST_k}}$ be $C_{i_{REST_k}}$. Depending on the actual execution time, $C_{i_{REST_k}}$ can be between zero at a minimum and $C_{i_{WCET}} - C_{i_{PET_k}}$ at a maximum. In the following, the worst-case scenario, $C_{i_{REST_k}} = C_{i_{WCET}} - C_{i_{PET_k}}$, is assumed.

$J_{i_{PET_k}}$ and $J_{i_{REST_k}}$ are given the absolute deadlines, $d_{i_{PET_k}}$ and $d_{i_{REST_k}}$, respectively, that are calculated by the following formulas. (In the formulas, $T_i$ is a period of $\tau_i$ and $U_i$ is the processor utilization by $\tau_i$ which is calculated using the corresponding WCET. That is, $U_i = C_{i_{WCET}}/T_i$.)

$$d_{i_{PET_k}} = k \times T_i + \frac{C_{i_{PET_k}}}{U_i} \qquad (2)$$

$$d_{i_{REST_k}} = d_{PET_k} + \frac{C_{i_{REST_k}}}{U_i} = (k+1) \times T_i \qquad (3)$$

After the deadlines are given, the two instances are scheduled based on EDF algorithm. From the formula (2) and (3), $J_{i_{PET_k}}$ is always executed before $J_{i_{REST_k}}$. If the execution of $J_{i_{PET_k}}$ finishes at or before the predicted execution time, $J_{i_{REST_k}}$ does not exist. ($C_{i_{REST_k}}$ is zero and therefore $J_{i_{REST_k}}$ is not executed. )

An example of the division of a task into two instances is shown in Fig. 2. The horizontal axis is time in ticks. A task, $\tau_i$, has the period of $T_i = 8$, and WCET of $C_{i_{WCET}} = 2$. The utilization by $\tau_i$ is 2/8 = 0.25. The fifth execution of $\tau_i$ is shown in the figure. For the fifth execution, the predictive execution

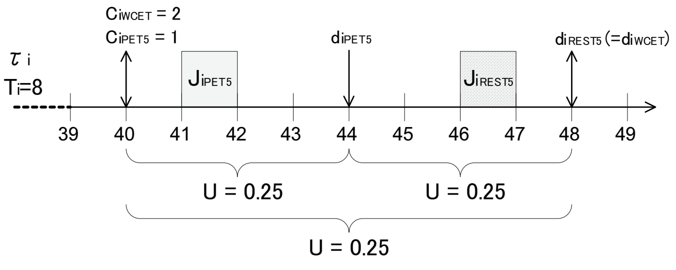


Fig. 2. Two instances from a task.

time of $J_{i_{PET_5}}$, $C_{i_{PET_5}}$, is assumed to be 1. Therefore, $d_{i_{PET_5}}$ becomes $40+1/0.25=44$. $J_{i_{PET_5}}$ starts at tick 41. When the execution spends $C_{i_{PET_5}}$, it is suspended if the task execution has not finished yet. Then, the remaining execution restarts at tick 46 in this example.

## C. Example of Adaptive EDF

In this section, an example of the adaptive EDF where the response time can be shorten is shown. In Fig. 3, the top figure (1) is the schedule of the original EDF and the bottom (2) is the schedule of the adaptive EDF. The horizontal axis is time in ticks. In both (1) and (2), two tasks, $\tau_1$ and $\tau_2$, have their period, 4 and 6, and their WCET, 2 and 2, respectively, which leads to the processor utilization of $U_1 = 2/4 = 0.5$ for $\tau_1$ and $U_2 = 2/6 = 0.33$ for $\tau_2$. Actual execution times of $\tau_1$ and $\tau_2$ are 2 and 1, respectively. (The actual execution times do not change in this example.) In this example, the importance of $\tau_2$ is assumed to be high and therefore $\tau_2$ is the target of the adaptive EDF. $\alpha$ for PET calculation is 0.5 in this example.

The original EDF gives the average response time of the first three instances of $\tau_2$ of (3 + 1 + 3) / 3 = 2.33. On the other hand, in the adaptive EDF, the first execution of $\tau_2$ has $C_{2_{PET_0}} = C_{2_{WCET}} = 2$. Therefore, $d_{2_{PET_0}} = 0 \times T_2 + (2/U_2) = 6$. The second execution has $C_{2_{PET_1}} = 0.5 \times 2 + 0.5 \times 1 = 1.5$ and $d_{2_{PET_1}} = 1 \times T_2 + 1.5/U_2 = 10.5$. For the third execution, $C_{2_{PET_2}} = 0.5 \times 1.5 + 0.5 \times 1 = 1.25$ and $d_{2_{PET_2}} = 2 \times T_2 + 1.25/U_2 = 15.75$. (The calculated deadlines are depicted with dashed arrows in the figure.) In the original EDF, the third execution starts at tick 14 and finishes at tick 15. In the adaptive EDF, it starts at tick 12 and finishes at tick 13 since the corresponding deadline is earlier than that for the fourth execution of $\tau_1$. Therefore, the average response time of the first three instances is (3 + 1 + 1) / 3 = 1.67, which is shorter than that in the original EDF.

## D. Adaptive EDF Schedulability

Schedulability of the adaptive EDF can be discussed as follows. By letting $U_i$ be the processor utilization (based on the WCET) by the most important periodic task $\tau_i$, the adaptive EDF is the same as Total Bandwidth Server (TBS) [4] with the server bandwidth of $U_i$. (TBS is a scheduling algorithm for mixed task sets of periodic and aperiodic tasks. In TBS, each aperiodic task is given deadline according to its WCET and the server bandwidth. After the deadline assignment, the whole task set is scheduled by EDF. )

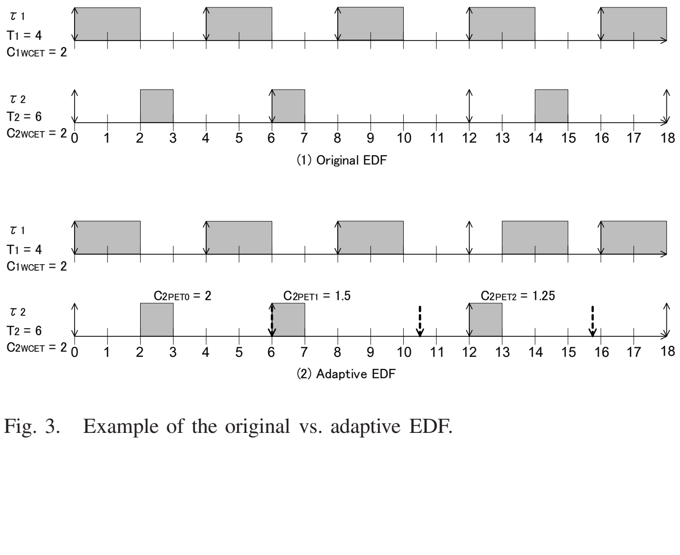


Fig. 3. Example of the original vs. adaptive EDF.

The two instances made of the periodic task are executed as aperiodic requests. The deadline assignment for the two instances leads to being the same as that in TBS. That is, when execution times of the two instances are $C_{i_{PET_k}}$ and $C_{i_{REST_k}}$, the deadlines given by TBS are equal to the formula (2) and (3). Therefore, the schedulability test for TBS in the literature [4] can be directly applied to the adaptive EDF, that is, if and only if the total processor utilization is equal to or smaller than 1 ($U \leq 1$), the whole task set is schedulable by EDF.

## E. Implementation Complexity

In the proposed algorithm, task execution is divided into two different instances. However, operating systems should manage a task with a single information set, task control block, although the two instances run separately. This is realized by re-setting up deadline and re-inserting the task in a ready queue when PET elapses but the task instance has not finished, which is the only difference from the original EDF. To find that execution reaches its PET, the scheduler should be executed every tick timing. This is achieved by calling the scheduler when timer/tick interrupts occur, which is a natural procedure that operating systems usually follow.

## III. Evaluation

In this section, effects of the proposed adaptive EDF are shown by comparing it with the existing EDF and RM in the simulations.

In the evaluation, four methods, (1) Original EDF, (2) the adaptive EDF, (3) the adaptive EDF with perfect prediction where execution times of all execution instances are known in advance (Oracle EDF), and (4) RM are compared. (Oracle EDF can be regarded as the ideal adaptive EDF, where PET is perfectly predicted for every instance execution and gets equal to the actual execution time, and therefore $C_{i_{REST_k}}$ is always zero.)

In the simulations, task sets consist of periodic tasks with the total CPU utilization ($U_p$) from 70% to 100% at intervals of 5%. $U_p$ is based on WCETs of all tasks. For each periodic task, its period is decided by uniform distribution between 0 and 100 ticks and its WCET is decided by uniform distribution between 1/10 and 1/3 of the period. Actual execution times of instances of each task is decided by uniform distribution between 1/3 and 1/1 of the WCET. (Execution instances of the same task have different execution times.) The observation period is 100,000 ticks.

For each $U_p$, ten different periodic task sets were simulated and the average values of response times are shown. For the proposed method, the weighting coefficient for PET calculation ($\alpha$ in the section II-A) is 0.5. In the simulations for all task sets, no deadline misses occurred even in RM, since actual processor utilization was lower than $U_p$ that was based on WCETs.

## A. Results

Fig. 4 shows average response times of the most important task when a task with the shortest period is regarded as the most important task. The horizontal axis indicates $U_p$, and the vertical axis indicates average response times of the target task normalized to the results of RM. In the figure, RM exhibited shortest response times. This is natural since it always gives the highest priority to the important task (with the shortest period). As a whole, it is confirmed that the adaptive EDF provides shorter average response times than the original EDF. When $U_p$ is 95%, the adaptive EDF reduced average response times of the important task by 8.3%. It is also confirmed that the oracle EDF approaches RM, which means that the adaptive EDF could exhibit almost as short response times as RM if the prediction of execution times was perfect.

Fig. 5 shows results when a task with the longest period is regarded as the most important task. The original, adaptive, and oracle EDF generated shorter response times than RM. This is because RM gives the lowest priority to the most important task (with the longest period). When $U_p$ is 100%, the adaptive EDF reduced average response times of the important task by 14.2%. From the results, it is shown that the adaptive EDF can shorten response times of the most important task especially when an application requires task's importance independent of its period.

## IV. Conclusion

In this paper, in the EDF environments, a scheduling method of shortening response times by dividing execution of an important task based on the predictive execution time and assigning an earlier deadline is proposed. This method is effective especially when an application task requires importance independent of its period. In other words, when a task set consists of hard tasks with short periods and soft tasks with relatively long periods, the proposed methods gives a reasonable solution for shortening response times of the soft tasks. In the simulation-based evaluation, average response times of an important task were reduced by 14.2% at a maximum.

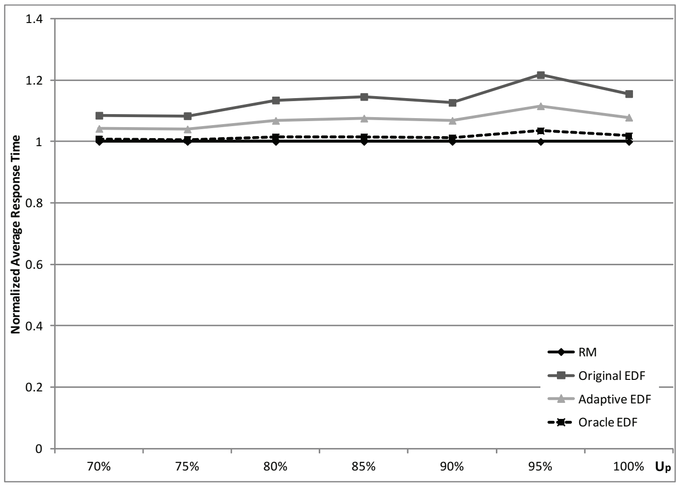


Fig. 4. Results – Target is the task with shortest period.

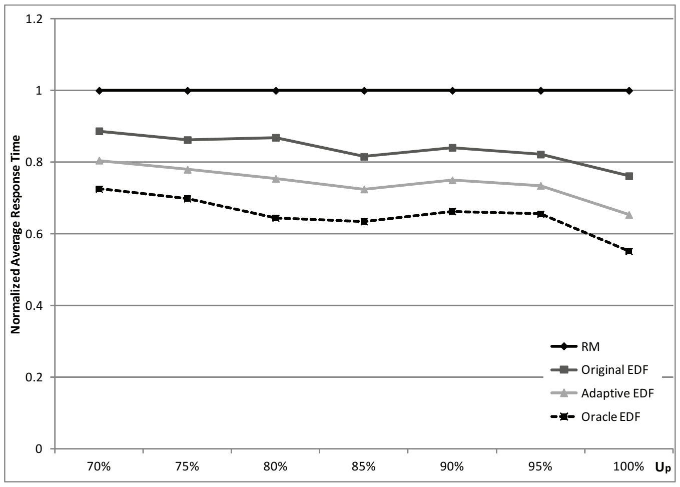


Fig. 5. Results – Target is the task with longest period.

Currently, obtaining PETs is simply based on the weighted average of the actual execution time of a previous instance and the previous PET, with the weighting coefficient $\alpha = 0.5$. If PETs can be perfectly predicted for every instance, the oracle EDF is achieved and further improvement can be expected. For example, there was room for further reduction in the average response times, by 28% in the evaluation. Therefore, more elaborate calculation/prediction methods of PETs need to be explored. In addition, the evaluation in this paper used task sets that were generated using probability distribution. To reflect actual situations where different instances of the same task spend different execution times, evaluation with actual program codes is desired. In the future, evaluation with actual program codes including scheduling overheads by operating systems should be performed.

There is another problem to be examined. Only a task is regarded as an important task in the evaluation. On the other hand, two or more tasks can be treated as important ones and be under the adaptive EDF control, although it is easily predicted that no gains cannot be obtained if all tasks are controlled as important tasks. Tradeoff between the number of important tasks and the amount of reduced response times should be evaluated.

## References

[1] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," Journal of the Association for Computing Machinery, Vol. 20, No. 1, pp. 46–61, January 1973.

[2] T. Lundqvist and P. Stenström, "Timing Anomalies in Dynamically Scheduled Microprocessors," Proc. of IEEE Real-Time Systems Symposium, pp. 12–21, December 1999.

[3] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, "The Worst-Case Execution Time Problem – Overview of Methods and Survey of Tools," ACM Trans. on Embedded Computing Systems, Vol. 7, No. 3, Article No. 36, April 2008.

[4] M. Spuri and G. C. Buttazzo, "Efficient Aperiodic Service under Earliest Deadline First Scheduling," Proc. of IEEE Real-Time Systems Symposium, pp. 2–11, December, 1994.